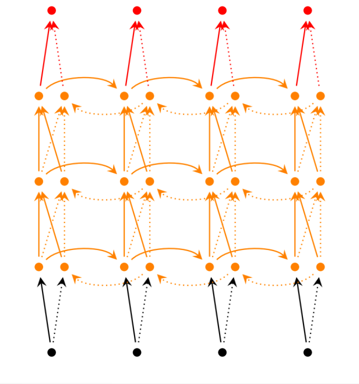
A Tutorial on PyTorch

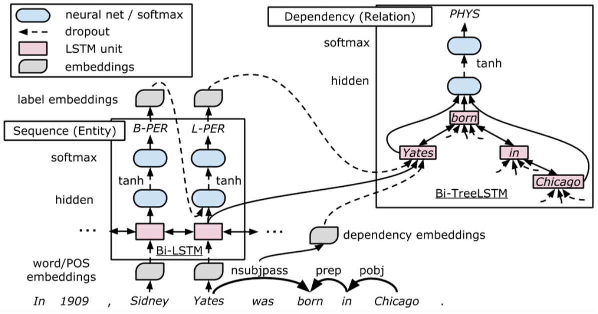
Speaker : Zhenyu Wu

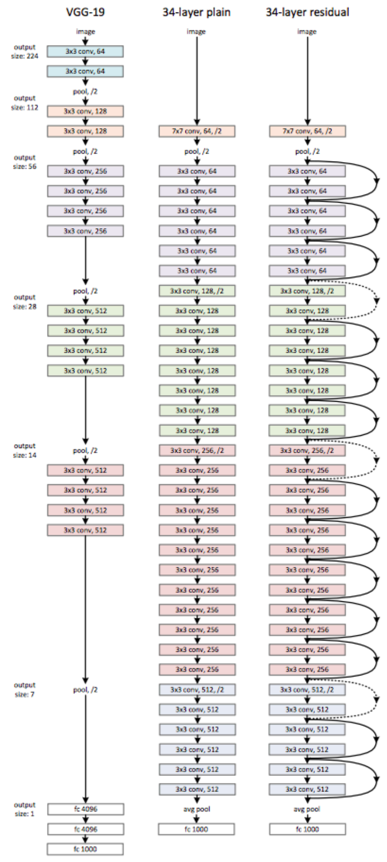
Why a DL Library is Necessary?

- Complicated DL archit
  - Easily build big comput graphs

il

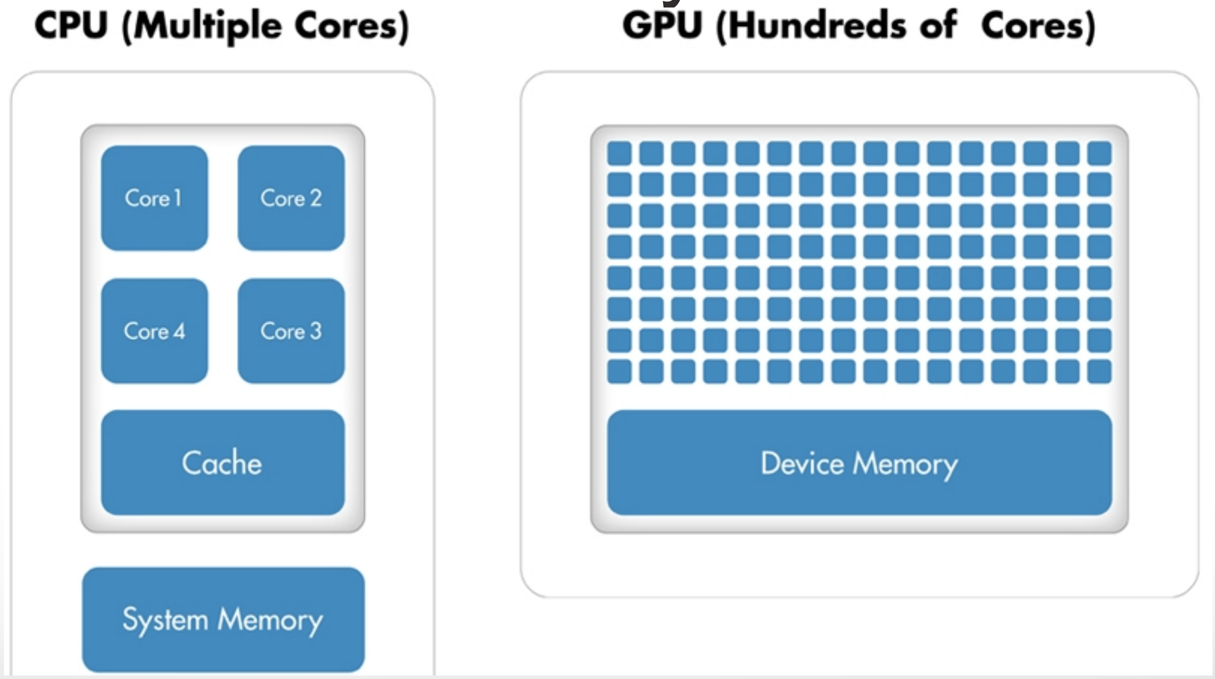






Why a DL Library is Necessary?

- Run it all efficiently on GPU



Popular Deep Learning Libraries

Caffe

Microsoft CNTK

dmlc mxnet

Caffe2

TensorFlow

Chainer

theano

torch

K

PYTORCH

PaddlePaddle

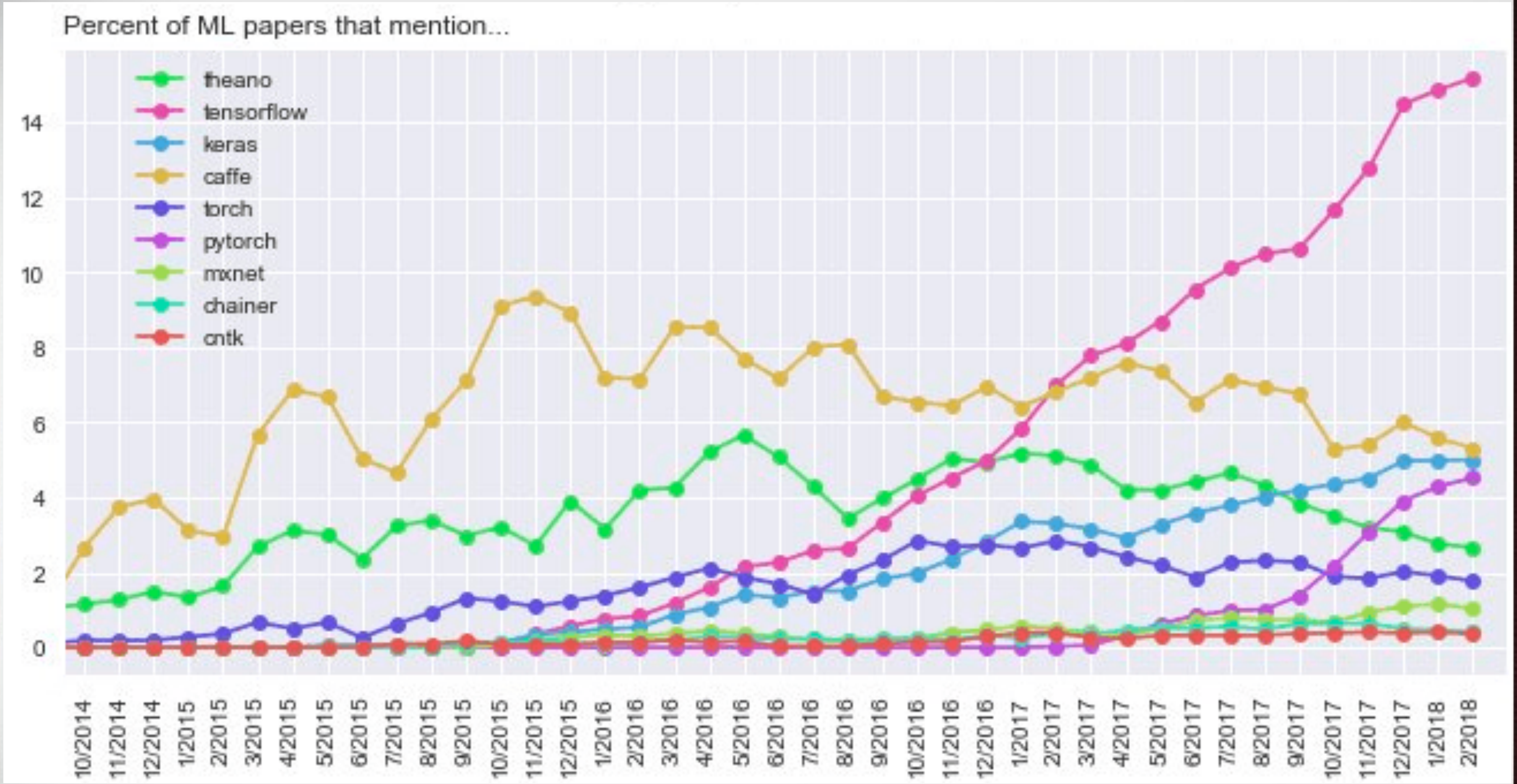
Keras vs PyTorch vs TensorFlow

Keras:

- High-level API
- On top of TensorFlow, CNTK, or Theano
- Easy to use
- Less flexible

PyTorch (Facebook) vs TensorFlow (Google):

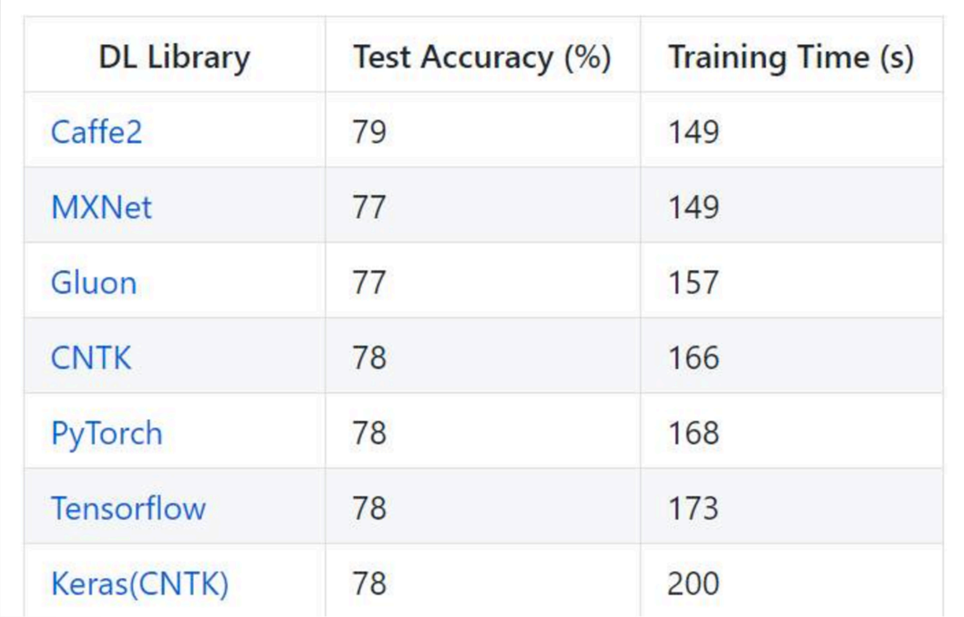
- Low-level API
- Flexible to write any complex models (research)
- Dynamic vs Static computation graphs



Percent of ML papers that mention...

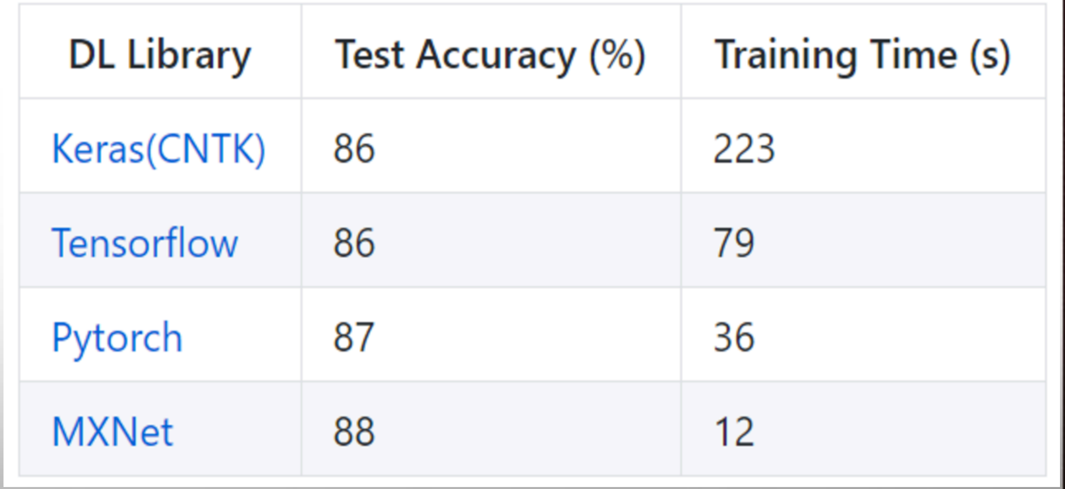
Performance Comparison

Test VGG on CIFAR-10



DL Library	Test Accuracy (%)	Training Time (s)
Caffe2	79	149
MXNet	77	149
Gluon	77	157
CNTK	78	166
PyTorch	78	168
Tensorflow	78	173
Keras(CNTK)	78	200

Test LSTM on IMDB
GPU-accelerated LSTM



DL Library	Test Accuracy (%)	Training Time (s)
Keras(CNTK)	86	223
Tensorflow	86	79
Pytorch	87	36
MXNet	88	12

Why PYTORCH

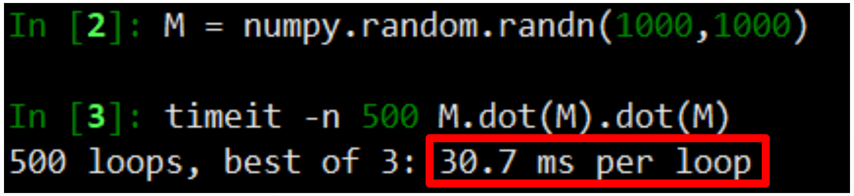
- Numpy-like Tensor Calculation
  - numpy.reshape() ➔ torch.view()
  - numpy.concatenate() ➔ torch.cat()
  - numpy.dot() ➔ torch.dot()
  - Support slicing, indexing, broadcasting
- Powerful tensor calculation with GPU support
- Flexible auto-differentiation & auto-grad system
- Dynamic Computation Graph (suitable for NLP research)
- Good community support and documentation
  - Latest deep learning models: GAN, VGG, ResNet, seq2seq

PyTorch as A Tensor Library

- Tensor operations: slicing, indexing, math operations, linear algebra, reductions
  - CPU & GPU
  - Fast! (comparison on speed of **matrix multiplication**)

$$\boldsymbol{M} * \boldsymbol{M} * \boldsymbol{M} \qquad M \in \mathbb{R}^{1000\times 1000}$$

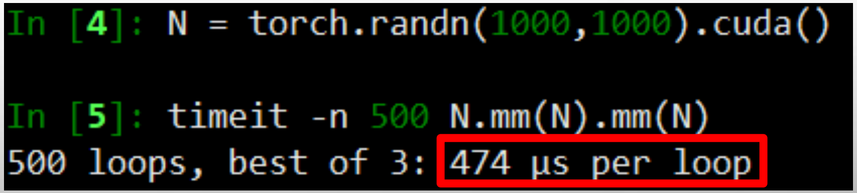
Numpy



```
In [2]: M = numpy.random.randn(1000,1000)

In [3]: timeit -n 500 M.dot(M).dot(M)
500 loops, best of 3: 30.7 ms per loop
```

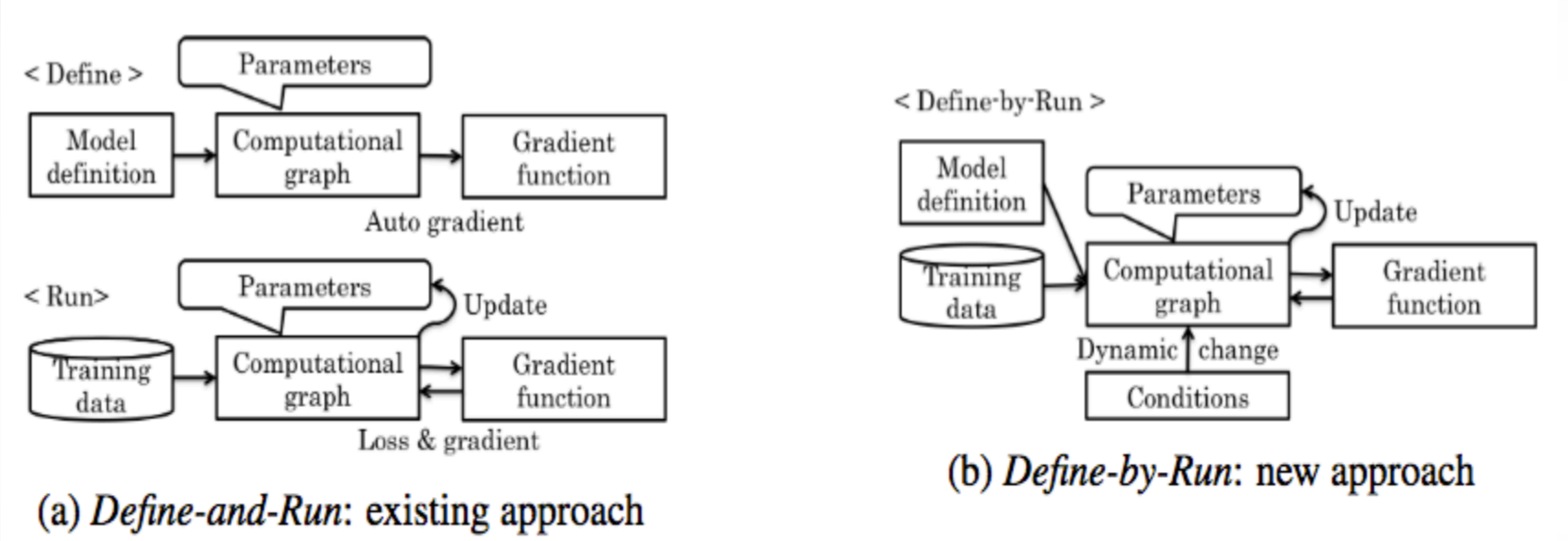
PyTorch



```
In [4]: N = torch.randn(1000,1000).cuda()

In [5]: timeit -n 500 N.mm(N).mm(N)
500 loops, best of 3: 474 µs per loop
```

Dynamic computation graph



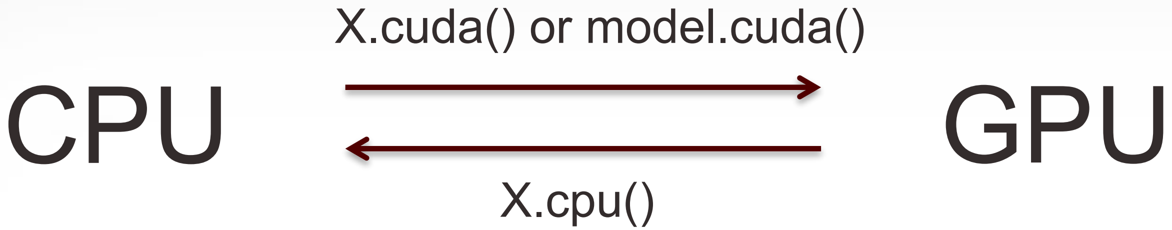
(a) *Define-and-Run*: existing approach

(b) *Define-by-Run*: new approach

Static graph

Dynamic graph

Using GPU with Pytorch



CPU

X.cuda() or model.cuda()

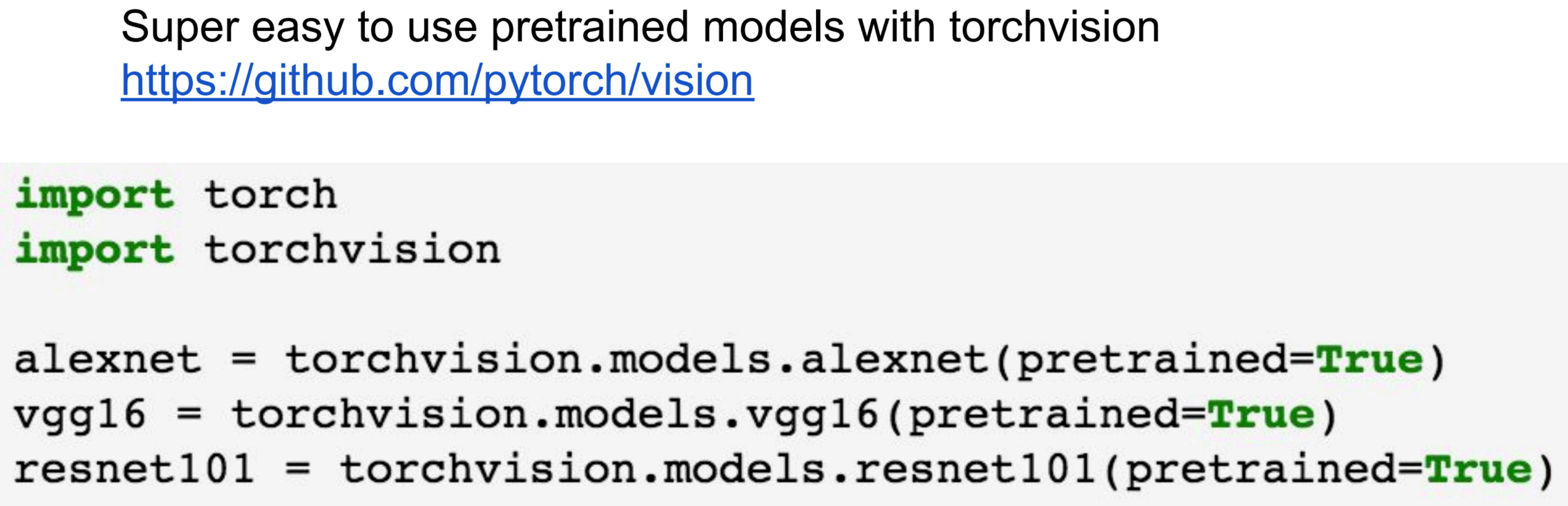
X.cpu()

GPU

- Use .cuda() on your data (stored in Variable)
- Use .cuda() on your model (stored in Module)

```
model = MyModel(…)
if torch.cuda.is_available():
    X.cuda()
    model.cuda()
```

Pretrained model in Pytorch



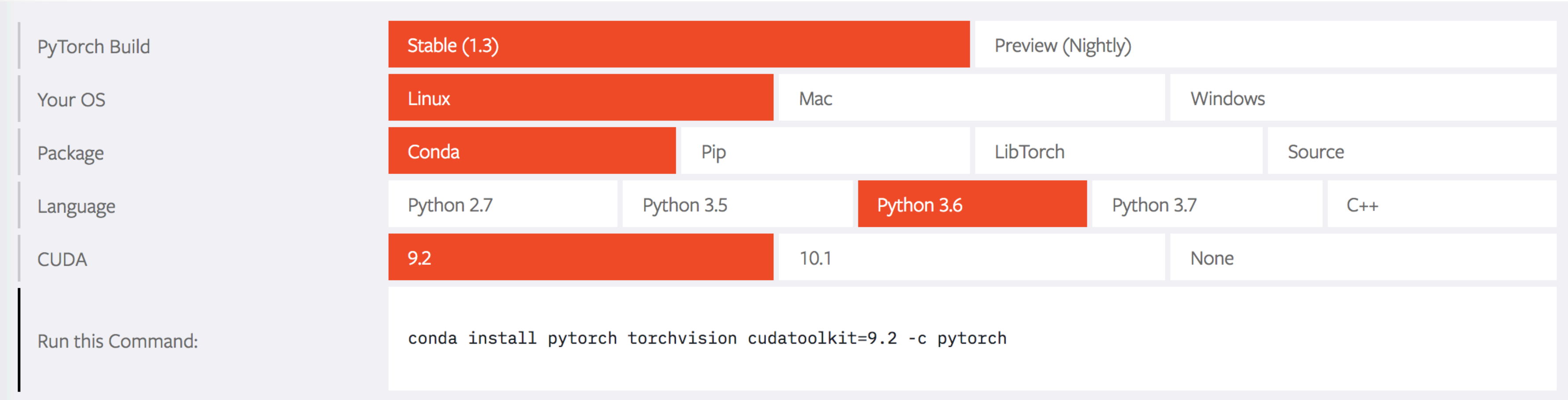
Super easy to use pretrained models with torchvision
https://github.com/pytorch/vision

```
import torch
import torchvision

alexnet = torchvision.models.alexnet(pretrained=True)
vgg16 = torchvision.models.vgg16(pretrained=True)
resnet101 = torchvision.models.resnet101(pretrained=True)
```

Installation

PYTORCH



PyTorch Build: [✓ Stable (1.3)] Preview (Nightly)

Your OS: [✓ Linux] Mac Windows

Package: [✓ Conda] Pip LibTorch Source

Language: Python 2.7 Python 3.5 [✓ Python 3.6] Python 3.7 C++

CUDA: [✓ 9.2] 10.1 None

Run this Command:

```
conda install pytorch torchvision cudatoolkit=9.2 -c pytorch
```

Thanks!

TEXAS A&M UNIVERSITY